Multimedia Manual User manual

发行版本 (HEAD detached at 9f3e22c)

SOPHGO

2024 年 07 月 25 日

Directory

CHAPTER 1

Disclaimer

Legal Disclaimer

Notice

The purchased products, services and features are stipulated by the contract made between SOPHGO and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied. The information in this document is subject to change

without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Technical Support

Address
Floor 6, Building 1, Yard 9, FengHao East Road, Haidian District, Beijing, 100094, China

Website
https://www.sophgo.com/

Email
sales@sophgo.com

Phone
+86-10-57590723 +86-10-57590724

CHAPTER 2

Release note

Version	Date of Release	Descriptions
V0.1.0	2023.08.31	First edition adds OpenCV private API and FFMPEG API
V0.2.0	2024.05.08	Second edition adds GStreamer API

CHAPTER 3

Install sophon-media

sophon-media offers different types of installation on different Linux distributions. Please select the corresponding according to your system way, do not mix multiple installation methods on one machine. "1.0.0" is used as an example in the following description, which is currently in practice The installation version is subject to change.

The following $arch $system is configured according to the actual architecture:

- The host is x86 CPU, $arch is AMD64, and $system is x86_64
- The host is ARM64 or Flying CPU, the $arch is ARM64 and the $system is ARC64

If you use a Debian/Ubuntu system: The sophon-media installation package consists of six files:

- sophon-media-soc-sophon-ffmpeg_1.0.0_arm64.deb
- sophon-media-soc-sophon-ffmpeg-dev_1.0.0_arm64.deb
- sophon-media-soc-sophon-opencv_1.0.0_arm64.deb
- sophon-media-soc-sophon-opencv-dev_1.0.0_arm64.deb
- sophon-media-soc-sophon-gstreamer_1.0.0_arm64.deb
- sophon-media-soc-sophon-gstreamer-dev_1.0.0_arm64.deb

Thereinto:

sophon-media-soc-sophon-ffmpeg/sophon-media-soc-sophon-opencv/sophon-media-soc-sophon-gstreamer contains ffmpeg/opencv/gstreamer runtime environments (library files, tools, etc.); sophon-media-soc-sophon-ffmpeg-dev/sophon-media-soc-sophon-opencv-dev/sophon-media-soc-sophon-gstreamer-dev contains the development environment (header files, pkgconfig, cmake, etc.). If you are only installing on a

deployment environment, you do not need to install sophon-media-soc-sophon-ffmpeg-dev/sophon-media-soc-sophon-opencv-dev/sophon-media-soc-sophon-gstreamer-dev.

sophon-media-soc-sophon-ffmpeg depends on the sophon-libsophon package, while sophon-media-soc-sophon-opencv depends on sophon-media-soc-sophon-ffmpeg, Therefore it is necessary in the installation sequence Install libsophon first, then sophon-media-soc-sophon-ffmpeg, and finally install sophon-media-soc-sophon-opencv. The package sophon-media-soc-sophon-ffmpeg only depends on the sophon-libsophon package, and there is no dependency relationship between sophon-media-soc-sophon-opencv and sophon-media-soc-sophon-ffmpeg.

The installation steps are as follows: INSTALLING LIBSOPHON DEPENDENCIES (SEE LIBSOPON USER MANUAL) [sudo dpkg -i sophon-soc-libsophon_0.4.9_arm64.deb]

Install plugin dependencies:

sudo apt install libgstreamer1.0-dev gstreamer1.0-plugins-base libgstreamer-plugins-base1.0-dev gstreamer1.0-plugins-bad libgstreamer-plugins-bad1.0-dev gstreamer1.0-gl gstreamer1.0-tools gstreamer1.0-plugins-good

Install sophon-media

If you use a **Debian/Ubuntu** system:

sudo dpkg -i sophon-media-soc-sophon-ffmpeg_1.0.0_arm64.deb

sudo dpkg -i sophon-media-soc-sophon-ffmpeg-dev_1.0.0_arm64.deb

sudo dpkg -i sophon-media-soc-sophon-opencv_1.0.0_arm64.deb

sudo dpkg -i sophon-media-soc-sophon-opencv-dev_1.0.0_arm64.deb

sudo dpkg -i sophon-media-soc-sophon-gstreamer_1.0.0_arm64.deb

sudo dpkg -i sophon-media-soc-sophon-gstreamer-dev_1.0.0_arm64.deb

Run the following command in the terminal, or logout and then log in the current user to use the installed tool:

source /etc/profile

Note: When in SOC mode, the system is already pre-installed:

sophon-media-soc-sophon-ffmpeg

sophon-media-soc-sophon-opencv

Just follow the steps above to install:

sophon-media-soc-sophon-ffmpeg-dev_1.0.0_arm64.deb

sophon-media-soc-sophon-opencv-dev_1.0.0_arm64.deb

sophon-media-soc-sophon-gstreamer_1.0.0_arm64.deb

sophon-media-soc-sophon-gstreamer-dev_1.0.0_arm64.deb

The installation location is:

```
/opt/sophon/
├─── libsophon-0.4.9
├─── libsophon-current -> /opt/sophon/libsophon-0.4.9
├─── sophon-ffmpeg_1.0.0
|   ├─── bin
|   ├─── data
|   ├─── include
|   ├─── lib
|   |   ├─── cmake
|   |   └─── pkgconfig
|   └─── share
├─── sophon-ffmpeg-latest -> /opt/sophon/sophon-ffmpeg_1.0.0
├─── sophon-opencv_1.0.0
|   ├─── bin
|   ├─── data
|   ├─── include
|   ├─── lib
|   |   ├─── cmake
|   |   ├─── opencv4
|   |   └─── pkgconfig
|   ├─── opencv-python
|   ├─── share
|   └─── test
├─── sophon-opencv-latest -> /opt/sophon/sophon-opencv_1.0.0
├─── sophon-gstreamer_1.0.0
|   ├─── data
|   ├─── include
|   ├─── lib
└─── sophon-gstreamer-latest -> /opt/sophon/sophon-gstreamer_1.0.0
```

The deb package installation method does not allow you to install multiple different versions of the same package, but you may use other methods to place several different versions

under /opt/sophon. When installing using the deb package, /opt/sophon/sophon-ffmpeg-latest, /opt/sophon/sophon-opencv-latest, and /opt/sophon/sophon-gstreamer-latest point to the last version installed. After uninstalling, it will point to the remaining latest version, if any.

The directories include, lib/cmake, lib/pkgconfig, and include are respectively created by the installation of the sophon-media-soc-sophon-ffmpeg-dev, sophon-media-soc-sophon-opencv-dev, and sophon-media-soc-sophon-gstreamer-dev packages.

Uninstall method:

If you use a Debian/Ubuntu system:

```
sudo apt remove sophon-media-soc-sophon-opencv-dev sophon-media-soc-sophon-opencv
sudo apt remove sophon-media-soc-sophon-ffmpeg-dev sophon-media-soc-sophon-ffmpeg
sudo apt remove sophon-media-soc-sophon-gstreamer-dev sophon-media-soc-sophon-gstreamer
or
sudo dpkg -r sophon-media-soc-sophon-opencv-dev
sudo dpkg -r sophon-media-soc-sophon-opencv
sudo dpkg -r sophon-media-soc-sophon-ffmpeg-dev
sudo dpkg -r sophon-media-soc-sophon-ffmpeg
sudo dpkg -r sophon-media-soc-sophon-gstreamer-dev
sudo dpkg -r sophon-media-soc-sophon-gstreamer
```

If you are using a different Linux system: The installation package consists of one file:

- sophon-media-soc_1.0.0_aarch64.tar.gz

You can install it by following these steps:

INSTALL THE LIBSOPHON PACKAGE ACCORDING TO THE LIBSOPHON USER MANUAL, AND THEN

tar -xzvf sophon-media-soc_1.0.0_aarch64.tar.gz

sudo cp -r sophon-media_1.0.0_$system/* /

sudo ln -s /opt/sophon/sophon-ffmpeg_1.0.0 /opt/sophon/sophon-ffmpeg-latest

sudo ln -s /opt/sophon/sophon-opencv_1.0.0 /opt/sophon/sophon-opencv-latest

sudo ln -s /opt/sophon/sophon-sample_1.0.0 /opt/sophon/sophon-sample-latest

sudo sed -i "s/usr/local/opt/sophon/sophon-ffmpeg-latest/g" /opt/sophon/sophon-ffmpeg-latest/lib/pkgconfig/*.pc

sudo sed -i "s/^prefix=.*$/prefix=/opt/sophon/sophon-opencv-latest/g" /opt/sophon/sophon-opencv-latest/lib/pkgconfig/opencv4.pc

Finally, install the bz2 libc6 libgcc dependency library (this part needs to choose the corresponding installation package according to different operating systems, which is not introduced here) Then some configuration work:

```
To add the library and executable path:
sudo cp /opt/sophon/sophon-ffmpeg-latest/data/01_sophon-ffmpeg.conf /etc/ld.so.conf.d/
```

(续下页)

(接上页)

```
sudo cp /opt/sophon/sophon-opencv-latest/data/02_sophon-opencv.conf /etc/ld.so.conf.d/
sudo ldconfig
sudo cp /opt/sophon/sophon-ffmpeg-latest/data/sophon-ffmpeg-autoconf.sh /etc/profile.d/
sudo cp /opt/sophon/sophon-opencv-latest/data/sophon-opencv-autoconf.sh /etc/profile.d/
sudo cp /opt/sophon/sophon-sample-latest/data/sophon-sample-autoconf.sh /etc/profile.d/
source /etc/profile
```

Uninstall method:

sudo rm -f /etc/ld.so.conf.d/01_sophon-ffmpeg.conf

sudo rm -f /etc/ld.so.conf.d/02_sophon-opencv.conf

sudo ldconfig

sudo rm -f /etc/profile.d/sophon-ffmpeg-autoconf.sh

sudo rm -f /etc/profile.d/sophon-opencv-autoconf.sh

sudo rm -f /etc/profile.d/sophon-sample-autoconf.sh

sudo rm -f /opt/sophon/sophon-ffmpeg-latest

sudo rm -f /opt/sophon/sophon-opencv-latest

sudo rm -f /opt/sophon/sophon-sample-latest

sudo rm -rf /opt/sophon/sophon-ffmpeg_1.0.0

sudo rm -rf /opt/sophon/sophon-opencv_1.0.0

sudo rm -rf /opt/sophon/sophon-sample_1.0.0

sudo rm -rf /opt/sophon/opencv-bmcpu_1.0.0

Precautions

- If you need to use the python interface of sophon-opencv, manually set the environment variables:

export PYTHONPATH=$PYTHONPATH:/opt/sophon/sophon-media_1.0.0/opencv-python

CHAPTER 4

Use sophon-sample

sophon-sample offers different types of installation on different Linux distributions. Please select the pair according to your system How should it be, do not mix multiple installation methods on one machine. The "1.0.0" in the following description is an example only, subject to current reality The installation version is subject to change.

The following $arch $system is configured according to the actual architecture:

- The host is x86 CPU, $arch is AMD64, and $system is x86_64
- The host is ARM64 or Flying CPU, the $arch is ARM64 and the $system is ARC64

If you use a Debian/Ubuntu system:

The sophon-sample installation package consists of the following files:

- sophon-media-soc-sophon-sample_1.0.0_arm64.deb

Thereinto:

- sophon-media-soc-sophon-sample includes several applications for testing sophon-ffmpeg/sophon-opencv;
- sophon-media-soc-sophon-sample relies on the sophon-ffmpeg/sophon-opencv package from the previous chapter.

The installation steps are as follows:

```
INSTALLING LIBSOPHON DEPENDENCIES (SEE LIBSOPON USER MANUAL)
Installation sophon-media (refer to the previous section)
Install sophon-sample
```

If you use a Debian/Ubuntu system:

- sudo dpkg -i sophon-media-soc-sophon-sample_1.0.0_arm64.deb

The installation location is:

```
/opt/sophon/
├── libsophon-0.4.9
├── libsophon-current -> /opt/sophon/libsophon-0.4.9
├── sophon-ffmpeg_1.0.0
├── sophon-ffmpeg-latest -> /opt/sophon/sophon-ffmpeg_1.0.0
├── sophon-opencv_1.0.0
├── sophon-opencv-latest -> /opt/sophon/sophon-opencv_1.0.0
├── sophon-gstreamer_1.0.0
├── sophon-gstreamer-latest -> /opt/sophon/sophon-gstreamer_1.0.0
├── sophon-sample_1.0.0
│   ├── bin
│   │   ├── test_bm_restart
│   │   ├── test_ff_bmcv_transcode
│   │   ├── test_ff_scale_transcode
│   │   ├── test_ff_video_encode
│   │   ├── test_ff_video_xcode
│   │   ├── test_ff_hw_bmcv_transcode
│   │   ├── test_ff_resize_transcode
│   │   ├── test_ff_bmjpeg
│   │   ├── test_ff_bmjpeg_dec_recycle
│   │   ├── test_ff_hw_bmcv_transcode
│   │   ├── test_ocv_jpubasic
│   │   ├── test_ocv_jpumulti
│   │   ├── test_ocv_vidbasic
│   │   ├── test_ocv_video_xcode
│   │   ├── test_ocv_vidmulti
│   │   ├── test_gst_transcode
│   │   └── test_gst_vcmulti
│   ├── data
│   └── samples
└── sophon-sample-latest -> /opt/sophon/sophon-sample_1.0.0
```

The deb package installation method does not allow you to install multiple different versions of the same package. But you can use other approaches Several different versions were placed under /opt/sophon. When installing using the deb package/opt/sophon/sophon-sample- latest will point to the last version installed. After uninstalling, it will point to the remaining latest version, if any.

Note: In SOC mode, the DEB installation package is

sophon-media-soc-sophon-sample_1.0.0_arm64.deb

The installation location is the same as above

Uninstall method:

If you use a Debian/Ubuntu system:

- sudo apt remove sophon-media-soc-sophon-sample or
- sudo dpkg -r sophon-media-soc-sophon-sample_1.0.0_arm64.deb

Use case introduction: test_bm_restart

This use case is mainly used to test the video decoding function and performance under the ffmpeg module, and supports multiplexing and disconnection reconnection functions. Users can monitor the decoding of video and code streams through use cases.

test_bm_restart [api_version] [yuv_format] [pre_allocation_frame] [codec_name] [sophon_idx] [zero_copy] [input_file/url] [input_file/url]

Parameter:

-api_version
Specifies the ffmpegAPI version used by the decoding process

0: Use the old version of the decoded avcodec_decode_video2 interface

1: Use the new version of the decoding avcodec_send_packet interface

2: Use av_parser_parse2' s API for packet capture

-yuv_format
whether to compress data,

0 Indicates that it is not compressed

1 Represents compression

-pre_allocation_frame
The number of cache frames allowed, up to a maximum of 64

-codec_name
Specify the decoder, you can select h264_bm/hevc_bm, no is not specified

-sophon_idx
In SOC mode, this option can be set arbitrarily (not null), and its value will be ignored

-zero_copy
SOC mode, 0 means Host memory is enabled, 1 means not enabled

-input_file_or_url
Enter the file path or bitstream address

e.g
test_bm_restart 1 0 1 no 0 0 ./example0.mp4 ./example1.mp4 ./example2.mp4

test_ff_bmcv_transcode

This example is primarily used to test the video transcoding capability and performance under the FFMPEG module. By calling ff_video_decode, ff_video_encode data types and functions in the use case to implement the transformation of decoding before encoding code process, This ensures the correctness of the decoding and encoding functions. At the same time, this use case can also test the transcoding performance under ffmpeg The program outputs the instant transcoding frame rate for reference.

test_ff_bmcv_transcode [platform] [src_filename] [output_filename] [encode_pixel_format] [codecer_name] [width] [height] [frame_rate] [bitrate] [thread_num]

Parameter:

-platform
Platform: SoC

-src_filename
Input a file name such as x.mp4 x.ts, and so on

-output_filename
Transcode output file names such as x.mp4, x.ts, etc

-encode_pixel_format
Encoding formats such as I420.

-encoder_name
Encoding h264_bm,h265_bm.

-width
Encoding width (32,4096)

-height
Encoding Height (32,4096)

-frame_rate
Encoded frame rate

-bitrate
Encoded bit rate encode bitrate 500 $<$ bitrate $<$ 10000

-thread_num
Number of threads

e.g

soc mode example:
test_ff_bmcv_transcode soc example.mp4 test.ts I420 h264_bm 800 400 25 3000 3

test_ff_scale_transcode

This use case is primarily used to test the functionality and performance of video transcoding under FFMPEG. This feature is implemented through a process of decoding and then encoding Now, the data types and functions in ff_video_decode, ff_video_encode are mainly called.

test_ff_scale_transcode [src_filename] [output_filename] [encode_pixel_format] [codecer_name] [height] [width] [frame_rate] [bitrate] [thread_num] [zero_copy] [sophon_idx]

Parameter:

-src_filename
Input a file name such as x.mp4 x.ts···

-output_filename
Output file names such as x.mp4 x.ts···

-encode_pixel_format
The encoding format is such as I420

-codecer_name
The code name is such as h264_bm,hevc_bm,h265_bm

-height
The encoding height

-width
Encoding width

-frame_rate
Encoded frame rate

-bitrate
Encoded bit rate

-thread_num
The number of threads used

-zero_copy
0: copy host mem,1: nocopy.

-sophon_idx
Device index

e.g

test_ff_scale_transcode example.mp4 test.ts I420 h264_bm 800 400 25 3000 3 0 0

test_ff_video_encode

This use case is primarily used to test the encoding capabilities of a video under the ffmpeg module. The video input is limited to I420 and NV12 formats. By calling this use case, users can get encapsulated video files, which are supported by FFMPEG.

test_ff_video_encode <input file> <output file> <encoder> <width> <height> <roi_enable> <input pixel format> <bitrate(kbps)> <frame rate>

Parameter:

-input_file
Input the video path

-output_file
Output video file name

-encoder
H264 or H265, default is H264

-width
Video width, output and input must be consistent,256 <= width <= 8192

-height
Video height, output and input must be consistent,128 <= height <= 8192

-roi_enable
Whether to enable ROI, 0 means no on, 1 means on

-input_pixel_format
I420(YUV, default), NV12.

-bitrate
Output bitrate, 10 < bitrate <= 100000, default frame rate x width x height/8

-framerate
Output frame rate, 10 < framerate <= 60, default is 30

e.g

- test_ff_video_encode <input file> <output file> H264 width height 0 I420 3000 30
- test_ff_video_encode <input file> <output file> H265 width height 0 I420
- test_ff_video_encode <input file> <output file> H265 width height 0 NV12
- test_ff_video_encode <input file> <output file> H265 width height 0

test_ff_video_xcode

This use case is primarily used to test the functionality and performance of video transcoding under FFMPEG. This feature goes through a process of decoding before encoding implementation, which mainly calls data types and functions in ff_video_decode,ff_video_encode. Transcoded video The resolution is the same as the original video, and the bitrate cannot exceed 10000kbps or less than 500kbps, otherwise it will be set as the default value 3000kbps。If the bitrate of the transcoded video is the same as the original video, the duration should also be the same. Some dropped frames are normal Phenomenon.

test_ff_video_xcode <input file> <output file> encoder framerate bitrate(kbps) isdmabuffer

Parameter:

-input_file
Input file

-output_file
Output file

-encoder
Encoder H264 or H265.

-isdmabuffer
Whether the memory is enabled consistently, 1 means not enabled, and 0 means enabled

e.g

- test_ff_video_xcode ./file_example_MP4_1920_18MG.mp4 tran5.ts H264 30 3000 1
- test_ff_video_xcode ./file_example_MP4_1920_18MG.mp4 tran5.ts H264 30 3000 0

test_ff_hw_bmcv_transcode

This use case is primarily used to test the functionality and performance of video transcoding under FFMPEG. This feature goes through a process of decoding before encoding implementation, which mainly calls data types and functions in ff_video_decode,ff_video_encode.

test_ff_hw_bmcv_transcode [platform] [src_filename] [output_filename] [codecer_name] [width] [height] [frame_rate] [bitrate] [thread_num] [en_bmx264] [zero_copy] [sophon_idx]

Parameter:

-platform
soc

-src_filename
Input file

-output_filename
Output file

-codecer_name
Encoder H264 or H265.

-width
Encode the video width

-height
Encode the video height

-frame_rate
Encoded frame rate

-bitrate
Encoded bit rate

-thread_num
thread num

-en_bmx264
soc : 0

-zero_copy
soc : 0

-sophon_idx
soc : 0

e.g

- test_ff_hw_bmcv_transcode soc INPUT.264 out.264 h264_bm 1920 1080 25 3000 0 0 0

test_ff_resize_transcode

This use case is primarily used to test the functionality and performance of video transcoding under FFMPEG. This feature goes through a process of decoding before encoding implementation, which mainly calls data types and functions in ff_video_decode,ff_video_encode.

test_ff_resize_transcode [src_filename] [output_filename] [encode_pixel_format] [codecer_name] [height] [width] [frame_rate] [bitrate] [thread_num] [zero_copy] [sophon_idx]

Parameter:

-platform
soc

-src_filename
Input file

-output_filename
Output file

-encode_pixel_format
The encoding format is such as I420

-codecer_name
Encoding names such as h264_bm, hevc_bm, h265_bm

-width
Encode the video width

-height
Encode the video height

-frame_rate
Encoded frame rate

-bitrate
Encoded bit rate

-thread_num
The number of threads

-zero_copy
0: copy host mem,1: nocopy

-sophon_idx
Device index

e.g

test_ff_resize_transcode example.mp4 test.ts I420 h264_bm 800 400 25 3000 3 0 0

test_gst_transcode

This use case is mainly used to test the functionality and performance of video transcoding under gstreamer. This feature goes through the process of decoding and then encoding Yes, support H.264, H.265, JPEG and other format inputs.

test_gst_transcode [video_path] [output_path] [dec_type] [enc_type] [bps] [gop] [gop_preset] [cqp] [qp_min] [qp_max] [q_factor]

Parameter:

-video_path
The path of the video file

-output_path
The path to save output transcode video file

-dec_type
h26x input codec format type 1:h264 2:h265 3:JPEG 4: decode bin chose by gst

-enc_type
h26x the output codec format type 1:h264 2:h265 3:JPEG

-bps
Encode target bitrate

-gop
h26x encode Group of pictures starting with I frame

-gop_preset
h26x encode a gop structure preset option [1-7]

-cqp
h26x encode constant quality mode set >=0 bsp invalid

-qp_min
h26x encode Min Quantization Parameter

-qp_max
h26x encode Max Quantization Parameter

-q_factor
jpeg quality parameter

e.g

test_gst_transcode -v 1920x1080.mp4 -d 4 -e 1 -g 50 -b 2000000 -o tc_case24.h264

test_gst_vcmulti

This use case is mainly used to test the frame rate, and internally uses videotestsrc to generate 1080p YUV test data, and outputs the frame rate of each channel. No bitstream or yuv will be output.

test_gst_vcmulti [video_path] [codec_type] [is_enc] [num_chl] [disp] [trans_type]

Parameter:

-video_path
The path of the video file

-codec_type
h26x codec format type 1:h264 2:h265 3:JPEG

-is_enc
0: decode multi test, 1: encode multi test, 2: transcode multi test

-num_chl
The numbers of codec channel

-disp
Display every channel fps

-trans_type
When is_enc is set to 2, the encode type: 1 for h264, 2 for h265, and 3 for JPEG

e.g

test_gst_vcmulti -c 1 -e 1 -n 4 -d 1

gst-launch-1.0

This use case is mainly used to test the functionality and performance of jpeg encoding under gstreamer.

e.g

gst-launch-1.0 filesrc location=640x480_420p.yuv ! rawvideoparse format=i420 width=640 height=480 ! bmjpegenc ! filesink location=case0.jpg

Detailed steps for pipelines:

filesrc
Read data from a file

rawvideoparse
Parse the original video data, specify the format as I420 (YUV420p), with a width of 640 and a height of 480

bmjpegenc
Encode video data into JPEG format using a BMJPEG encoder

filesink
Write the encoded JPEG data into a file

gst-launch-1.0

This use case is mainly used to test the functionality and performance of jpeg decoding under gstreamer, and calls the bmdec plugin to achieve hardware acceleration of video decoding.

e.g

gst-launch-1.0 filesrc location=JPEG_1920x1088_yuv420_planar.jpg ! jpegparse ! bmdec ! filesink location=case12.yuv

Detailed steps for pipelines:

filesrc
Read data from a file

jpegparse
Parse JPEG data

bmdec
Decode JPEG data using BM decoder

filesink
Write the decoded YUV data into a file

gst-launch-1.0

This test case is mainly used to test the functionality and performance of H.265 video encoding under gstreamer.

e.g

gst-launch-1.0 filesrc location=1080p_nv12.yuv ! rawvideoparse format=nv12 width=1920 height=1080 ! bmh265enc gop=50 ! filesink location=case0.h265

Detailed steps for pipelines:

filesrc
Read data from a file

rawvideoparse
Parse the original video data, specify the format as NV12, width as 1920, and height as 1080

bmh265enc
Use the BM H.265 encoder to H.265 encode the video and set the GOP to 50

filesink
Write the encoded H.265 video data into a file

gst-launch-1.0

This test case is mainly used to test the functionality and performance of H.264 video decoding under gstreamer.

e.g

gst-launch-1.0 filesrc location=1920x1080.mp4 ! qtdemux ! h264parse ! bmdec ! video/x-raw,format=NV12 ! filesink location=case1.nv12

Detailed steps for pipelines:

filesrc
Read data from a file

qtdemux
Decompose the elements of QuickTime format files to extract audio and video streams from them

h264parse
Analyze H.264 data

bmdec
Decode video data using BM decoder

video/x-raw,format=NV12
Convert the decoded video data to NV12 format

filesink
Write the converted NV12 video data into a file

gst-launch-1.0

This use case is mainly used to test the functionality and performance of video transcoding under gstreamer, which is achieved through the process of decoding first and then encoding.

e.g

gst-launch-1.0 -e filesrc location=1920x1080.mp4 ! qtdemux ! h264parse ! bmdec ! bmh265enc gop=50 bps=1000000 qp-min=24 qp-max=45 gop-preset=5 ! filesink location=case4.h265

Detailed steps for pipelines:

filesrc
Read data from a file

qtdemux
Decompose the elements of QuickTime format files to extract audio and video streams from them

h264parse
Analyze H.264 data

bmdec
Decode video data using BM decoder

bmh265enc
Use the BM H.265 encoder to H.265 encode the video, set GOP to 50, bit rate to 1000000, minimum quantization parameter to 24, and maximum quantization parameter to 45, GOP preset value is 5

filesink
Write the encoded H.265 video data into a file

gst-launch-1.0

This use case is mainly used to test the functionality and performance of VPSS color conversion under gstreamer.

e.g

gst-launch-1.0 filesrc location=1920x1080_yuv420p.bin blocksize=3110400 num-buffers=1 ! ‘video/x-raw, format=(string)I420, width=(int)1920,

height=(int)1080, framerate=(fraction)1/1' ! bmvpss ! 'video/x-raw, format=(string)RGB, width=(int)1920, height=(int)1080, framerate=(fraction)1/1' ! filesink location=vpss_case01.bin

Detailed steps for pipelines:

filesrc
Read data from the file, set blocksize (size of read data block) and num buffers (number of read data packets)

video/x-raw, format=(string)I420, width=(int)1920, height=(int)1080, framerate=(fraction)1/1
The input data is in I420 format, specifying width, height, and frame rate

bmvpss
Use the BMVPSS plugin to process video streams

video/x-raw, format=(string)RGB, width=(int)1920, height=(int)1080, framerate=(fraction)1/1
The processed data is in RGB format, with specified width, height, and frame rate

filesink
Write the processed RGB format data into a file

gst-launch-1.0

This use case is mainly used to test the functionality and performance of VPSS scaling under gstreamer.

e.g

gst-launch-1.0 filesrc location=2048x2048_rgb.bin blocksize=12582912 num-buffers=1 ! 'video/x-raw, format=(string)RGB, width=(int)2048, height=(int)2048, framerate=(fraction)1/1' ! bmvpss ! 'video/x-raw, format=(string)RGB, width=(int)16, height=(int)16, framerate=(fraction)1/1' ! filesink location=vpss_case15.bin

Detailed steps for pipelines:

filesrc
Read data from the file, set blocksize (size of read data block) and num buffers (number of read data packets)

video/x-raw, format=(string)RGB, width=(int)2048, height=(int)2048, framerate=(fraction)1/1
The data is in RGB format, with specified width, height, and frame rate

bmvpss
Use the BMVPSS plugin to process video streams

video/x-raw, format=(string)RGB, width=(int)16, height=(int)16, framerate=(fraction)1/1
The processed data is in RGB format, with specified width, height, and frame rate

filesink

Write the processed RGB format data into a file

CHAPTER 5

Develop with sophon-media

After installing sophon-media, users can link sophon-media libraries to their compilers in two ways Middle.

** If you use the Make compilation system **

It is recommended that users use pkgconfig to find the sophon-media library.

In previous installations, we added Sophion-Media' s pkgconfig path to environmental variation PKG_CONFIG_PATH. Therefore, users can add the following statement to the Makefile:

```
CFLAGS = -std=c++11

# add libsophon dependency because sophon-ffmpeg rely on it
CFLAGS += -I/opt/sophon/libsophon-current/include/
LDFLAGS += -L/opt/sophon/libsophon-current/lib -lbmcv -lbmlib -lbmvideo -lbmvpuapi -
↪lbmvpulite -lbmjpuapi -lbmjpulite -lbmion

# add sophon-ffmpeg
CFLAGS += $(shell pkg-config --cflags libavcodec libavformat libavfilter libavutil libswscale)
LDFLAGS += $(shell pkg-config --libs libavcodec libavformat libavfilter libavutil libswscale)

# add sophon-opencv
CFLAGS += $(shell pkg-config --cflags opencv4)
LDFLAGS += $(shell pkg-config --libs opencv4)
```

You can then use the sophon-ffmpeg and sophon-opencv libraries in the makefile.

Note: When the system has another copy of ffmpeg or opencv installed under /usr/lib or /usr/local/lib, Be careful to check that the correct sophon-ffmpeg/sophon-opencv path is

found. If the search is not correct Indeed, you need to explicitly specify the header file location and library file location

** If you use CMake to compile the system **

Users can add the following statement to CMakeLists.txt:

```
# add libsophon
find_package(libsophon REQUIRED)
include_directories(${LIBSOPHON_INCLUDE_DIRS})
link_directories(${LIBSOPHON_LIB_DIRS})

# add sophon-ffmpeg
set(FFMPEG_DIR /opt/sophon/sophon-ffmpeg-latest/lib/cmake)
find_package(FFMPEG REQUIRED NO_DEFAULT_PATH)
include_directories(${FFMPEG_INCLUDE_DIRS})
link_directories(${FFMPEG_LIB_DIRS})

# add sophon-opencv
set(OpenCV_DIR /opt/sophon/sophon-opencv-latest/lib/cmake/opencv4)
find_package(OpenCV REQUIRED NO_DEFAULT_PATH)
include_directories(${OpenCV_INCLUDE_DIRS})

add_executable(${YOUR_TARGET_NAME} ${YOUR_SOURCE_FILES})

target_link_libraries(${YOUR_TARGET_NAME} ${FFMPEG_LIBS} ${OpenCV_LIBS})
```

The functions in sophon-ffmpeg and sophon-opencv can be called in the user' s code

```
#include <opencv2/opencv.hpp>

int main(int argc, char const *argv[])
{
    cv::Mat img = cv::imread(argv[1]);
    return 0;
}
```